Virtual Environments for Training

Quarterly Status Report (10)

January 1 - March 31, 1998

Submitted by R. Stiles
Lockheed Martin Advanced Technology Center

Prepared for
Office of Naval Research
Contract N00014-95-C-0179
April 1998
(CLIN 0004, CDRL A001 Progress Report)

Abstract: This report describes the Lockheed-Martin VET team efforts and accomplishments during the ninth quarter of the contract. Activity is reported for each of the software components of the Training Studio: VIVIDS, Steve, and Vista, as well as domain development and evaluation study. This report contains material submitted by Dr. Allen Munro at USC/BTL and Dr. Lewis Johnson at USC/ISI.

LOCKHEED MARTIN

Lockheed Martin Missiles & Space
Advanced Technology Center
3251 Hanover Street, Palo Alto, California 94304-1191

In reply refer to:
LMMS-DB/209-98
10 July 1998

Program Officer
Office of Naval Research
Code 342 - Helen Gigley
800 North Quincy Street
Arlington, VA 22217-5660

Subject: Contract No. N00014-95-C-0179 "Virtual Environments for Training"

Reference: (a) CLIN 0004, CDRL A001 "Progress Report"

Enclosure: (1) Progress Report covering the period 1/1/98 - 3/31/9897 (1 hard copy & 1 diskette)

Dear Dr. Gigley:

The report required per Reference (a) is provided as Enclosure (1).

Should you have questions or require additional information, please contact me at 650/424-2006, or by fax at 650/424-3330. I can also be reached by email at davina.brown@lmco.com

Very truly yours,

Davina Brown

Davina Brown
Contract Specialist

cc: Director, Naval Research Laboratory, Attn: Code 2627
Washington, DC 20375 (w/ 1 copy of enclosure)

✓Defense Technical Information Center
8725 John Kingman Road, Suite 0944
Fort Belvoir, VA 22060-6218 (w/ 2 copies of enclosure)

DCMC/Lockheed Martin, B/107, Sunnyvale, CA (w/ 1 copy of enclosure)

1. Summary

This quarter's efforts concentrated on the productization of the Training Studio. This report describes the efforts of Lockheed Martin Advanced Technology Center (ATC), USC/ISI, and USC/BTL for the Virtual Environments for Training contract during the period from January 1 - March 31, 1998.

The LM ATC focused on extensions to the Vista Viewer that facilitate immersive team and individual interactions, focusing on improving usability of Vista interaction services. ISI continued research and development on their pedagogical agent, Steve, focusing on motor control, tutorial feedback, authoring task improvements, and productization. The students working on the AASERT grant associated with VET made progress on their projects connected with the project. At BTL, research and development during this quarter have centered on three areas; improvements in the integration of a VET-featured VIVIDS with the other VET components, including Vista, autonomous agents, TrishTalk, and the VET sound server. enhancing the Gas Turbine Engine (GTE) control system simulation, and productizing the prototype VIVIDS for improved operation in VET systems.

2. Introduction

This report describes the efforts of Lockheed Martin, USC/ISI, and USC/BTL for the Virtual Environments for Training contract during the period from January 1 - March 31, 1998. This report discusses the design, development, and implementation of Training Studio software components: Vista Viewer, VIVIDS, and Steve, as well as publication and presentations explaining our work.

The purpose of our work is to explore, develop, and evaluate novel techniques for incorporating automated instruction in virtual environments. USC/ISI's focus has been on incorporating pedagogical capabilities in an intelligent agent architecture called Steve. We are investigating the following hypotheses: 1) that an agent architecture and knowledge representation can be developed that permits autonomous agents to act as guides, mentors, and team members, 2) that machine learning and high level languages can be employed to assist instruction developers in creating agent-based instruction, and 3) virtual environment technology enables new types of interactions between trainees and instructional systems, which improve the quality of instruction provided by the instructional systems.

BTL has demonstrated the correctness of the hypothesis that the 2D behavior authoring interface of RIDES can be adapted and extended to provide an effective and natural way to specify simulations for virtual environment training. The prototype authoring system for building simulation behaviors and structured tutorials for virtual environments is called VIVIDS (Virtual Interactive Intelligent Tutoring System Development Shell). Building on that success, work in this phase of the contract has, to some extent, shifted to preliminary productization—ensuring the robustness, completeness, and the openness of the prototype authoring and delivery system.

The VIVIDS authoring system constitutes the first system for *authoring* (as opposed to *programming*) robust complex interactive simulations for virtual environments. Furthermore, these authored simulations have features that support the near-automatic construction of certain types of structured tutorials. The combination of productive simulation authoring with efficient tutorial development is designed to make feasible the application of virtual environment technologies to a very wide range of technical training requirements. Extensions to the original system permit collaborations with pedagogical agents and support team training in virtual environments.

The Lockheed Martin team extended the Vista Viewer capabilities for human-computer interaction in a networked, real-time immersive training environment, continued work to optimize the Vista

software, and supported the development requirements of USC colleagues at ISI and BTL . Efforts toward productization of the Training Studio were increased during this period.

3. Methods, Assumptions & Procedures

We have been conducting a number of research investigations, each of which is directed at one or more of the objectives mentioned in the introduction. For each of the three system components, these investigations are conducted by one or more members of the primary research team, in collaboration with the other VET project participants.

Lockheed Martin's approach focuses on providing those capabilities that accomplish communications and scene display and manipulation for Steve and VIVIDS, as well as optimizing human interactions within the virtual environment. New capabilities are developed, tested, and released in a fast cycle to collaborating VET organizations for further evaluation and critique. Other capabilities are developed in response to a direct request by one of the other team members; or provided as a solution to a problem encountered by one of the collaborators.

The Lockheed Martin team members for the VET project are: Randy Stiles (Program Manager), Sandeep Tewari, Mihir Mehta, and Laurie McCarthy.

During the first quarter of 1998, the USC/ISI team consisted of the following individuals: Dr. Lewis Johnson (Principal Investigator), Dr. Jeff Rickel (research scientist), Mr. Marcus Thiebaux (programmer). The project was also assisted by Richard Angros (a graduate student), Ben Moore, and Anna Romero (undergraduate students), all working on the AASERT grant associated with the VET project. Moore took a leave from the project in January in order to study full time, he is expected to rejoin the project in May. Romero joined the project in January, and will continue through May.

USC/ISI research methodology is as follows. We identify a new capability that, if incorporated into Steve, would contribute to validating one of our research hypotheses. We then design a set of extensions to the Steve system that implements the capability. We develop a prototype implementation of the capability, and conduct a series of demonstrations and in-house tests. We then make arrangements for further evaluation of the capabilities by ourselves or our partner organizations.

During the second year of this project, the USC/BTL team for the VET project consisted of the following individuals: Dr. Allen Munro (Principal Investigator), Dr. Quentin Pizzini, and David Feldon.

Our methodology has been to progressively adapt VIVIDS to provide appropriate simulation and instruction services for a virtual environment delivered by Vista, to provide services to the Steve autonomous agent, and to exploit appropriately the speech (TrishTalk) and sound capabilities of the VET environment. These new capabilities are tested by developing large simulations and instruction materials using the revised authoring tools. Two levels of formative evaluation are pursued: both the usability of the revised authoring system and the functionality of the tutorials it produces must be examined. Based on, first, USC/BTL in-house evaluations , and, after initial revisions, the evaluations of our research partners—at Lockheed-Martin, at USC/ISI, and at the U.S. Air Force Laboratory, further modifications are made, and the tool-development, authoring and testing cycle resumes.

4. Results and Discussion

This section covers the results accomplished during this reporting period and discusses the significance of this work in terms of the VET project goals as well as contributions to respective research communities at large.

4.1 Software Development

Lockheed Martin, USC/BTL, and USC/ISI each accomplished major milestones regarding development of their respective components: Vista Viewer, VIVIDS, and Steve.

4.1.1 Simulation-based Training

This section describes the research and development efforts with respect to the VIVIDS component, focusing on: improvements in the integration of a VET-featured VIVIDS with the other VET components; enhancing the Gas Turbine Engine (GTE) control system simulation; improving the immersed student interface, and 'productizing' the prototype VIVIDS for improved operation in VET systems.

4.1.1.1 Improvements in the integration of VIVIDS.

VIVIDS has been improved to collaborate more effectively with Vista, Steve, TrishTalk, and the ISI sound server.

Improved Initialization

In the past, starting up a full VET environment was somewhat cumbersome. After starting Vista, VIVIDS, and all the other components, a number of steps had to be followed to read in the model files used and to initialize the simulation. It was difficult to tell when initialization had been completed, so that the system was ready for tutorial use. Now, a VIVIDS 2D scene provides a one-step graphical user interface (GUI) for model loading and simulation initialization. The user selects a file to be loaded. Visual feedback is provided, showing what files are in the process of being loaded. After all the needed models have been loaded, initialization is automatically triggered. Only when the initialization process has been completed does the 2D GUI return to its default state, showing that the system is ready for use.

Utilization of the Sound Server

VIVIDS can now make use of the sound server. A behaving object can start and stop sounds when appropriate, and can specify how close a student must be to hear the sound associated with the object. Both user-initiated sounds (such as clicks associated with switch throws or button presses) and ongoing environmental sounds (such as a repeating drip) can be initiated.

Configuration Services

Rich Angros at ISI is developing an automatic plan authoring system that makes use of a VIVIDS simulation as a controllable world for conducting experiments on the outcomes of actions. In order to experiment successfully, it is necessary to be able to return to an initial state before attempting an alternative sequence of actions.

VIVIDS has long had the notion of a *configuration*, a snapshot of the state of a complete simulation. Configurations have been used to ensure that a tutorial in a complex simulation system begins in a known state. We have provided the functionality of opening VIVIDS configuration control to other applications. This makes it possible for Angros's automatic Plan authoring system to ask VIVIDS to define new configurations and to install them, when appropriate.

Exporting Object Name Correspondences

In the VET system, there can be several different names for what is, from the perspective of a human or an agent user, a single object. There can be a name for the Vista model node that represents the object, and there can be two VIVIDS names: one a unique identifier for internal references, and the other a public "display name" for use with students.

Autonomous agents need to know about these name correspondences. As an aid to our colleagues who are developing such agents, the author's top view in VIVIDS now has a new command on its File menu: *Produce 3D Name File*. This command walks through the internal data of a VIVIDS simulation and produces a file that lists every object that has a corresponding Vista model node, together with the Vista name and the public display name of the object, which should be used during instruction.

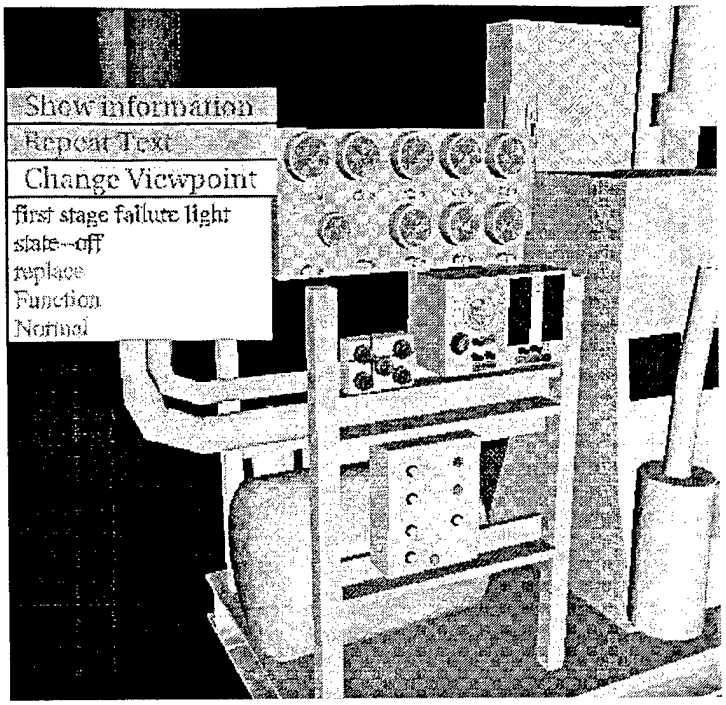


Figure 1 . Text Command Interface

4.1.1.2 Enhancing the Gas Turbine Engine (GTE) control system simulation.

The VIVIDS simulation has been brought into better compliance with the Vista 3D models. Several potential name conflicts were identified and eliminated, and certain behavior errors, which showed up when a user tested control panel lamps, were corrected.

In addition, many simulation objects were given associated sounds. These include control panel buttons, certain valves, and the sound of a leaking pipe. The ISI sound server was utilized, under the control of VIVIDS simulation objects.

An improved path authoring interface was developed, utilizing Vista's *viewpoint* capability. An instructional author can specify a path that the student will follow as he or she is moved from one point to another in the simulated world. Figure 2 below, shows the path authoring interface in use.

4.1.1.3 Improving the Immersed Student Interface

The immersed graphical student control interface has been modified to make it more understandable and more extensible.

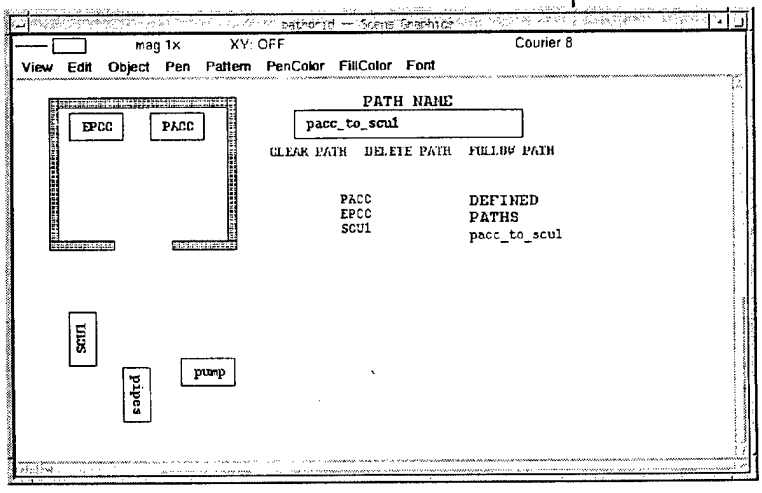


Figure 2. The Path Authoring Interface

Textual Command Interface for Students

The student command GUI, which formerly made use of icons with meanings not immediately evident to novices, has been replaced with a textual command interface, shown in Figure 1. Before, a fixed palette of three icons was utilized. Now, the command interface is a modifiable text menu.

Two new student commands

Two new commands have been added to the interface shown in Figure 1, above. *Repeat Text* makes TrishTalk repeat the last content-based text utterance. Vista viewpoints are linked to each other in a closed list. The *Change Viewpoint* command takes the student to the next defined viewpoint defined in the Vista model.

4.1.1.4 'Productizing' VIVIDS for Improved Operation in VET Systems.

Several types of efforts have been carried out to better 'productize' VIVIDS as a component of VET.

Team Training support

Several instructional primitives that had not yet been made sensitive to *participant*. have been modified so that those types of instructional items can be directed to particular participants during team training.

Robust performance

In earlier versions of VIVIDS, if instruction attempted to speak when TrishTalk was not available, VIVIDS would hang. Now VIVIDS is alert to TrishTalk's presence, and it does not attempt speech when TrishTalk is absent.

Simplifications of design features for collaboration

A smaller number of special-purpose VIVIDS attributes are now required for collaborating with other VET components, such as Vista, Steve, TrishTalk, and the sound server. We anticipate that this simplification will improve the maintainability and adaptability of the system.

Speech buffer clearing

When an author uses a *ClearText* instruction item, it is the author's intention that recently presented textual materials should no longer be available. In the immersed 3D context, where pedagogical text is presented with speech-to-text, this instructional item now clears the speech buffer so that the Repeat Text command is no longer enabled for the student.

Opportunistic instruction improvements

In the preliminary release of opportunistic instruction, only very simple opportunistic lesson fragments could be delivered. Now any arbitrary lesson or lesson fragment can be presented using the opportunistic instruction mechanism.

Documentation

A sixty-two page set of preliminary documentation on authoring and utilizing VIVIDS-based simulations and tutorials for VET has been developed.

4.1.2 Pedagogical Agent Development

This section relates improvements to the Steve Pedagogical Agent in the use of dialog-centered speech, motor control in a complex graphical setting, Steve's graphical representation, and task authoring. It also relates ISI efforts in developing a sound server.

4.1.2.1 Motor Control

During this quarter, we extended Steve to include control over his facial expressions. Thiebaux completed a new VRML model of Steve's head that supports movable eyes, lips, eyebrows, and eyelids. (Previously, only the eyes were movable. Together, Rickel and Thiébaux designed an appropriate API to allow Steve to control his facial expressions, Thiebaux implemented the API, and Rickel integrated the API into Steve. Although the API supports a wide variety of facial expressions, Steve currently only makes use of a few: he blinks continuously, and he switches from his neutral expression to a talking face when speaking. However, Rickel is supervising an undergraduate student, Anna Romero, whose senior project focuses on supporting other facial expressions that would be useful in a tutorial context, and we hope to integrate such expressions into Steve's behavior soon.

In addition to our work on facial expressions, we improved Steve's motor control in several other ways. We improved Steve's use of gaze as he moves from object to object in the environment. We added the ability for an author to customize where Steve stands relative to objects he is manipulating; such knowledge is optional, but it can override Steve's default when necessary. For improved believability, Steve now gazes randomly around the room when not engaged in a task. Finally, Thiebaux has made good progress on arms for Steve, as well as more sophisticated use of hands, and we hope to finish that work and integrate it into Steve's behavior early in the next quarter.

4.1.2.2 Tutorial Feedback

Until recently, Steve only provided feedback to students when they asked questions. During this quarter, we extended Steve to provide feedback to students whenever they perform actions. Currently, Steve's feedback is very simple; he nods in agreement when the student takes an appropriate action, and says "no" while shaking his head when the student performs an inappropriate action. However, Steve has all the knowledge required to provide more elaborate feedback on the reasons why the student's action is inappropriate, so we are now in a position to extend Steve in that direction.

4.1.2.3 Authoring and Knowledge Acquisition

Richard Angros prepared his tutorial authoring system (Diligent) for usability evaluation. He improved the user interface, particularly those aspects concerned with editing plans, so that it would be possible to compare the effectiveness of Diligent's learning and experimentation techniques against an unassisted plan editor. He prepared a user's manual for the system, developed an evaluation plan, and conducted preliminary usability trials.

The initial trials revealed some problems with the tutorial materials. As expected, some users had difficulty in forming an overall picture of what they needed to do in order to author a procedure in Diligent. The tutorial manuals have since been corrected, and further usability trials are planned for the near future.

4.1.2.4 Preparing for Field Use

As the VET project draws to a close, our focus is shifting towards making Steve robust enough for eventual field use at the end of the project. We made good progress in that direction this quarter. In addition to making Steve more usable, Rickel has plugged up many of the holes in Steve's implementation that could result in undesirable behavior in atypical circumstances. Such work will continue to be a high priority for the remainder of the VET project.

4.1.2.5 Domain Development

We continued our application of Steve to team training for the operators of gas turbine engines. Specifically, we added knowledge to allow Steve to choose collision-free paths through the engine room, and we added tasks for the engine room operator during a loss of fuel oil pressure scenario. This individual checks all the suction valves for leaks, closing any leaky valves and ensuring that all others are wide open. At the end of this development, Rickel supported Lockheed as they prepared a video tape involving the full scenario involving five agents and multiple students.

4.1.3 Virtual Environment Interaction

During this quarter, Vista development centered on supporting evolving needs for immersive single and multi-participant interactions as well as to support requirements of the other Training Studio components.

4.1.3.1 Notification of VRML File Loading

In the previous versions of Vista it was hard to find out if a VRML file had finished loading. This issue was addressed during this quarter; results were implemented in a new release of Vista. Vista sends out a message of the form :

vrFinishedVRMLLoad all <url>

after the file has been loaded. The **<url>** is the URL for the file which was loaded. However, this message is sent out only if delayed inline loading has been disabled. By default delayed inline loading is disabled, i.e., inlines are loaded right away. Delayed inline loading can be disabled/enabled via the following Tscript message:

vrSwitchDelayedInline <participant> <value>

<value> can be 1 (ON) or 0 (OFF)

This capability is used by Vivids and Steve to find out when a particular model that they had requested Vista to load actually got loaded.

4.1.3.2 Adding VRML Files to existing ones

In the previous versions of Vista it was not possible to add VRML files to existing ones. Every time a VRML file was loaded it would replace the previous one. This was a serious limitation for Steve and Vivids as it prevented them from loading new VRML models on the fly without having to replace the existing VRML file. The Vista component was enhanced to add the capability to allow VRML files to be added to existing VRML files in Vista. The Tscript message used to accomplish this is:

vrAddVRMLFile <participant> <parent> <url>

This Tscript message allows the file pointed to by **<url>** to be loaded under the specified **<parent>**. Scoping of node names for files loaded this way is not yet done. The user has to make sure that there are no duplicate names between the files loaded this way and previously loaded VRML files in Vista.

After a file is loaded via **vrAddVRMLFile** notification is sent out through the following Tscript message:

vrAddedVRMLFile all <url>

This Tscript message is sent out only if delayed inline loading is disabled.

4.1.3.3 Changes to Viewpoint Node

A new exposedField "immersedRelease" was added to the Viewpoint node. This is not part of the VRML standard. This field is used to signal if interpolation needs to be done from a viewpoint's orientation to the user's orientation when the user arrives at a particular viewpoint while he/she is immersed. This field is of type SFBool. Its default value is TRUE which means that interpolation is done from the viewpoint's orientation to the user's orientation during immersed interaction. If set to FALSE then after the user arrives at a particular viewpoint his/her view just snaps from the viewpoint's orientation to their orientation. Since this is an exposedField it can be set from outside via the vrSetVRMLField Tscript message. This capability gives flexibility to users and lets them decide what kind of behavior they need when they arrive at a viewpoint while they are immersed.

Also, in this release of Vista when the user arrives at a particular viewpoint notification is sent out via a Tscript message of the form:

vrReachedViewpoint all <view_position> <view_orientation>

<view_position> and <view_orientation> are the position and orientation of the viewpoint the user just arrived at.

This capability is very useful if Steve or Vivids want to give a guided tour to users by moving them from one viewpoint to the other. Through the above Tscript message Vivids or Steve (or any other piece of software which has registered for this message) can find out as to when the user reached a particular viewpoint.

4.1.3.4 Bug fixes for Text capability

In the past, the Text capability in Vista would get really slow with repeated use. This problem was researched and is now fixed. The bottlenecks were found to be memory allocation and loading fonts used for displaying text in Performer. The text capability has been optimized by reducing memory allocations and font loading. Currently work is under way to incorporate

the VRML 2.0 text node as part of the VRML Browser capabilities of Vista.

4.1.3.5 General Utilities

It is now possible to query Vista for a particular node by giving its name. This is useful for finding out if a particular node is already part of the scenegraph. The Tscript message set up for this purpose has the following form:

vrFindNode <nodeName>

In response to this Tscript message Vista sends a Tscript message of the from:

vrFindNodeResult all <nodeName> <value> [nodeType]

<value> could be 1 or 0 depending on if the node was found or not. If the node was found then <nodeType> contains the type of the node which is the name of the Performer class, for e.g., pfDCS, pfGroup, etc.

The latest release of Vista also provides a way to move objects relative to other objects. The Tscript messages set up to accomplish this are:

vrTranslateRelative <target-dcs> <source-dcs> <x> <y> <z>

vrRotateRelative <target-dcs> <source-dcs> <h> <p> <r>

The first message, **vrTranslateRelative,** translates the <target-dcs> relative to the <source-dcs> by <x>,<y>,<z>. This is useful if the user wants to translate a object relative to another object without being concerned about where these two objects are in the scenegraph. A special value for **<source-dcs>** is "vrScene" which means that the translation is done in world coordinates.

The message **vrRotateRelative** rotates the **<target-dcs>** relative to the **<source-dcs>** by **<h>,<p>, <r>.** This is useful if the user wants to rotate a object relative to another object without being concerned about where these two objects are in the scenegraph. A special value for **<source-dcs>** is "vrScene" which means that the rotation is done in world coordinates.

4.1.4 Productization Efforts

Productization is a major task during Option 2 of the VET contract and underlies the design and development of the system extensions during this last phase. The development team members are developing or refining interfaces to each of Vista, Vivids, and Steve components to facilitate authoring. These efforts are described in the separate descriptions of each component in sections 4.1.1 - 4.1.3 of this report. Formal organization of component documentation is being gathered, revised, and assembled into a single reference manual. This manual will provide both the descriptions and instructions of Training Studio use and operations.

4.2 Meetings

The VET development team met twice during this quarter of the contract. The first meeting was held at ISI in Marina del Rey on January 28. This was a planning meeting, intended as the formal kickoff and milestone discussion and scheduling for Option 2 of the contract. Attendees were: Stiles, Tewari, Mehta, Johnson, Rickel, Thiebaux, Munro, Pizzini, Craig Hall (AFHRL), and Carol Horwitz (AFHRL). A second meeting was held at Lockheed Martin in Palo Alto on March 12 and focused on software integration and testing. This working meeting provided an opportunity to identify and work out any issues surrounding the integration of the latest versions of each software component. The integration meeting was attended by Stiles, Tewari, McCarthy, Munro, Pizzini, Johnson, and Rickel.

4.3 Presentations and Publications

Rickel and Johnson prepared several papers during this quarter. Their paper on Steve for the journal Applied Artificial Intelligence (a special issue on animated interface agents) was accepted for publication; they will complete the revisions by early May, and we will include a copy of the paper with our next quarterly report. Their video on Steve was accepted for presentation at the Second International Conference on Autonomous

Agents; the video, which was sent earlier to ONR, will not be revised, but we have included the accompanying written description with this quarterly report. They also had a paper accepted to the AAAI workshop on Multi-Modal Human-Computer Interaction, and their proposal to include Steve in the Intelligent Systems Demonstration Program at AAAI was also accepted. A paper on Steve appeared in *SIGART Bulletin* 8(1) this quarter, and Steve was also featured in a paper by Clark Elliott to appear in AI Magazine. Steve will also be presented in several upcoming conferences. Rickel has been invited to give presentations at Virtual Humans 3 in June and at Virtual Reality in Education and Training in London in July; Johnson has been invited to present at the Interaction Agents workshop in L'Aquila, Italy in May.

Johnson, Rickel, and Dr. Stacy Marsella presented a paper on team training in virtual environments to the Western Simulation Multi-Conference. Marsella and Johnson also submitted a paper on team training to the Intelligent Tutoring Systems conference; this paper was accepted as well.

Tewari presented a paper "Adapting VRML for Free-form Immersed Manipulation", which he co-authored with Stiles, Mehta, and McCarthy. The paper was well received. The conference also provided an opportunity to interact with other people working in the areas of VRML, Virtual Reality and 3D graphics.

Two videos were completed this quarter to facilitate demonstration of current Training Studio capabilities. The videos are not stand-alone productions, but rather, verbally annotated clips of the current system capabilities as illustrated in an immersed demonstration scenario. The decision was made to prepare short videos of system capabilities at development milestones as opposed to completing a formal updated video production of the VET program. This allows the team to maintain a more current demonstration resource, requiring a substantially lower budget than a formal production would require. The original VET production completed July 1997 presents the background and basic description of the Training Studio; and can be viewed in series with the latest demonstration footage for full understanding of the program. One of the videos, demonstrating VRML techniques developed for accomplishing maintenance tasks in a ship engineroom setting, was shown as part of Tewari's VRML 98 presentation.

5. Conclusions

This quarter marked the initiation of Option 2 of the contract and the beginning of a full productization effort by all members of the development team. Particular focus has been set on developing tools and interfaces to facilitate authoring of instructional scenarios within the Training Studio. Improvements to the user interfaces and interactivity continue, including facilitating conversations Another major effort, continued from previous quarters, but becoming more critical, is the optimization of the components. The team members have worked together to test all Training Studio components, identifying and addressing problem areas or bugs in their own or in their development partners' software.

Work in the upcoming quarter will continue the productization effort. At Lockheed Martin, we will continue support of our development partners providing functionality as requirements are identified. Progress will continue to complete the training scenarios for both individual and team tasks within the chosen casualty control domain. Planned extensions to the Profiler will be implemented to facilitate the launching of multiple Training Studio components from a single application.

ISI's work in the second quarter of 1998 will focus on several directions. Marcus Thiebaux will complete his work on arm and hand control for Steve, and Rickel will extend Steve to exploit these new capabilities. We plan to package up Marcus's code to allow other VET components, such as VIVIDS, to control their own animated bodies. Anna Romero will complete her senior project involving facial expressions for Steve, and

Rickel will integrate these into Steve's behavior. Rickel and Thiebaux will continue making their code more robust for eventual field use at the end of the VET project, and Rickel will prepare Steve's authoring interface and its associated documentation for eventual use by other course authors. Finally, Rickel and Johnson plan to conduct informal evaluations of Steve and begin design of more formal experiments.

In the coming quarter, USC/BTL efforts will be primarily directed toward these goals:

- Develop a substantive example of opportunistic instruction in the context of the Gas Turbine Engine (GTE) VET system.
- Enhance and refine the GTE simulation and develop new GTE tutorials.
- Automatically make use of the user's login name as the name of the Participant. This will make it unnecessary to manually set up the participant name for a tutorial.
- In collaboration with our research partners, assemble a complete sample course on the GTE.
- When the feature is made available to VIVIDS, take advantage of ISI's new accessible 'Steve body' to make use of the agent avatar in structured courses. Use the avatar to indicate objects, especially indicators, and to demonstrate actions.

6. Symbols, Abbreviations, and Acronyms

AAAI	American Association for Artificial Intelligence
AFHRL	U.S. Air Force Human Resources Laboratory
BTL	Behavioral Technologies Laboratories, located in Redondo Beach, CA, a performing organization in the Lockheed Martin VET effort, a laboratory of the University of Southern California.
COTR	Contracting Office Technical Representative. The Program Manager or Program Officer from the funding agency who provides technical direction for the program.
DARPA	Defense Advanced Research Projects Agency
DIS	Distributed Interactive Simulation, a real-time distributed message protocol used in training and operational simulations developed by ARPA and now an International Standards Organization standard.
FBM	Fleet Ballistic Missiles program, a Lockheed Martin program funded by the U.S. Navy for the production of submarine-launched ballistic nuclear missiles
GTE	Gas Turbine Engine - similar to jet engine, which drives propulsion of a Navy Ship. In our case we are usually referring to the LM2500 Gas Turbine Engine on USS Arleigh Burke (DDG-51) ships.
HPAC	High Pressure Air Compressor, an oil-free air compressing system prevalent on many navy vessels, which prepares compressed air for gas turbine engines.
ICAI	Intelligent Computer Aided Instruction, a method of instruction whereby an intelligent model of a student's understanding is used to guide a student during instruction using a computer.
IPEM	Integrated Planning, Execution and Monitoring architecture for coordinating different planning strategies as required for SOAR activities.
IRL	Institute for Research on Learning
ISI	Information Sciences Institute in Marina del Rey, CA, a performing organization in the Lockheed Martin VET effort, affiliated with the University of Southern California in Los Angeles, CA.
JPL	Jet Propulsion Laboratories, a National Laboratory affiliated with the National Aeronautics and Space Administration.
MCO	Multi-Channel Option for Silicon Graphics Onyx Workstations, a necessary option to provide separate video channels used in immersive virtual environment displays.
ONR	Office of Naval Research, the funding agency for the VET effort.
RIDES	Rapid Instructional Development for Educational Simulation
SIGART	Special Interest Group on Artificial Intelligence
SGI	Silicon Graphics Incorporated, a workstation company whose whole culture centers around fast 3D graphics.
SOAR	A platform independent, cognitive architecture based on a production system which seeks to address those capabilities required of a general intelligent agent.
STEVE	SOAR Training Expert for Virtual Environments
Tcl/Tk	A windowing interface toolkit assembled around a UNIX-shell like interpreter originally developed at UC Berkeley.
TScript	Training Script message protocol for virtual environments
URL	Uniform resource locator, a tag that indicates a media format and location on the Internet as part of the World Wide Web.
USC	The University of Southern California.
VE	Virtual Environment, a 3D visual display and accompanying simulation which represent some aspect of an environment. Expanded forms of VE also address other senses such as audio, touch, etc.
VET	Virtual Environments for Training, a Defense Department focused research initiative concerned with applying virtual environment technology to training
VR	Virtual Reality *see Virtual Environment*
VRIDES	Virtual Rapid Instructional Development for Educational Simulation. A special version of the RIDES program for use in developing simulations and tutorials that collaborate with Vista Viewer and Soar to deliver training in virtual environments.
VIVIDS	*See VRIDES above*
VRML	Virtual Reality Modeling Language, an analog to HTML used for documents, but focused on 3D objects and scenes for the World Wide Web.
WWW	World-Wide Web, a system incorporating the HTTP message protocol and the HTML document description language that allows global hypertext over the Internet.

Appendix A

STEVE: A Pedagogical Agent for Virtual Reality

To be presented at
Second International Conference on Autonomous Agents

May 10-13, 1998
Minneapolis/St. Paul, MN

STEVE: A Pedagogical Agent for Virtual Reality

Jeff Rickel and W. Lewis Johnson
Information Sciences Institute & Computer Science Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292-6695
rickel@isi.edu, johnson@isi.edu
http://www.isi.edu/isd/VET/vet.html

Abstract

We are exploring the use of virtual reality for training people how to perform tasks, such as operating and maintaining complex equipment. This video describes Steve, an agent we are developing that assists in the training. Steve is an autonomous, animated agent that cohabits the virtual world with students. Steve continuously monitors the state of the virtual world, periodically manipulating it through virtual motor actions. His objective is to help students learn to perform physical, procedural tasks. He can demonstrate tasks, explaining his actions, as well as monitor students performing tasks, providing help when they need it. In addition to teaching students individual tasks, he can also help them learn to perform multi-person team tasks: he can serve as a tutor for a student learning a particular role in the team, and he can play the role of a teammate when a human teammate is unavailable. By integrating previous work in agent architectures, intelligent tutoring systems, and computer graphics, Steve illustrates a new breed of computer tutor: a human-like agent that can interact with students in a virtual world to help them learn.

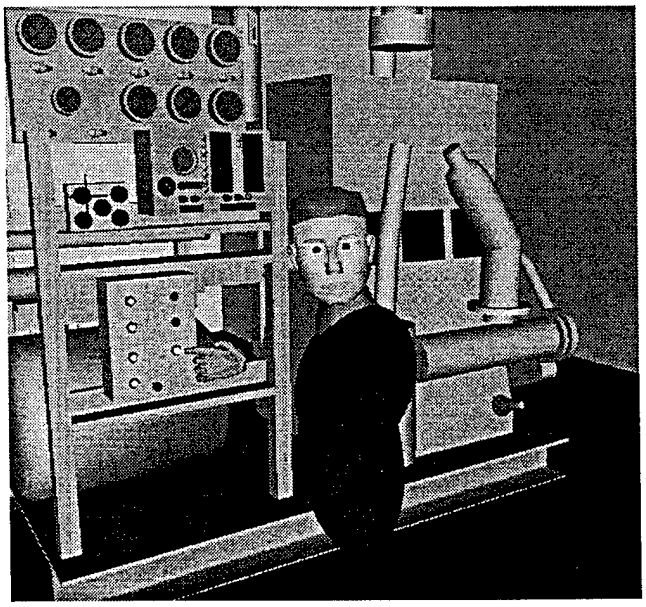


Figure 1: Steve describing a power light

1 Introduction

To master complex, real-world tasks, such as operating complicated machinery, people need hands-on experience facing a wide range of situations. They also need a mentor that can demonstrate procedures, answer questions, and monitor their performance, and they may need teammates if their task requires multiple people. Since it is often impractical to provide such training on real equipment, we are exploring the use of virtual reality instead; the training takes place in a three-dimensional, interactive, simulated mock-up of the student's work environment (as shown in the first video clip). Since mentors and teammates are often unavailable when the student needs them, we are developing an autonomous, animated agent that can play these roles. His name is Steve (Soar Training Expert for Virtual Environments).

Steve integrates methods from three research areas: intelligent tutoring systems, computer graphics, and agent architectures. This novel combination results in a unique set of capabilities. Steve has many pedagogical capabilities one would expect of an intelligent tutoring system. For example, he can answer questions such as "What should I do next?" and "Why?". However, because he has an animated body, and cohabits the virtual world with students, he can assist in ways that previous disembodied tutors cannot. For example, he can demonstrate actions, he can use gaze and gestures to direct the student's attention, and he can guide the student around the virtual world. Thus, virtual reality allows more human-like interactions among synthetic agents and humans than desktop interfaces can. Moreover, Steve's agent architecture allows him to robustly handle a dynamic virtual world, potentially populated with people and other agents; he continually monitors the state of the virtual world, always maintaining a plan for completing his current task, and revising the plan to handle unexpected events. All these capabilities are illustrated in the second video clip.

2 Steve's Architecture

Like many other autonomous agents that deal with a real or simulated world, Steve consists of two components: the first, implemented in Soar [5], handles high-level cognitive processing, and the second handles sensorimotor process-

Appears in the Proceedings of the Second International Conference on Autonomous Agents, Minneapolis/St. Paul, ACM Press, May 1998.

ing. The cognitive component interprets the state of the virtual world, carries out plans to achieve goals, and makes decisions about what actions to take. The sensorimotor component serves as Steve's interface to the virtual world, allowing the cognitive component to perceive the state of the world and cause changes in it. It monitors messages from the simulator describing changes in the state of the world, from the virtual reality software describing actions taken by the student and the student's position and field of view, and from speech recognition software describing the student's requests and questions posed to Steve. The sensorimotor module sends messages to the simulator to take action in the world, to text-to-speech software to generate speech, and to the virtual reality software to control Steve's animated body. Because the sensorimotor component provides a mapping between high-level action commands sent from the cognitive module and their realization in the virtual environment, it controls Steve's appearance in the virtual world. We have experimented with several graphical representations for Steve; in this video, Steve appears as a head and torso, with a hand that can manipulate and point at objects, as shown in Figure 1.

3 Team Training

We have recently extended Steve to understand team tasks, which require the collaboration of multiple people. To do this, we extended Steve's representation of tasks to distinguish the roles of different team members and we generalized all his pedagogical capabilities to use this extended representation. The third video clip shows two Steve agents demonstrating their ability to work together as a team. Their understanding of the team task shows in their anticipation and awareness of each other's actions.

Steve's ability to understand team tasks is important because it allows him to help people learn team tasks. In such training, Steve agents can play two valuable roles: they can serve as a tutor for an individual human team member, and they can play the role of missing team members, allowing students to practice team tasks without requiring all their human team members. The fourth video segment demonstrates this sort of training. A student practices his role in a two-person task while one Steve agent serves as his teammate and another Steve agent serves as his tutor.

4 Status and Future Work

Steve is fully implemented and integrated with the other software components on which he relies (i.e., virtual reality software, a simulator, and commercial speech recognition and text-to-speech products). We have tested him on a variety of Naval operating procedures; he can operate several consoles that control the engines aboard Naval ships, and he can perform an inspection of the air compressors on these engines. (The video shows simplified versions of some of these tasks.) We are continuing to extend his knowledge of these and related tasks. However, he is not limited to this domain; he can provide instruction in a new domain given only the appropriate declarative domain knowledge.

We have a variety of plans for future work. As an alternative to manual entry of domain knowledge, we are extending Steve to learn from demonstrations [1, 3]. To increase Steve's ability to motivate students, we are extending him to include emotions [2]. To allow him to express emotions, and to extend his range of nonverbal communication, we are giving him control over his facial expressions. We are adapting Steve's architecture for use in distance learning over the World Wide Web [4]. Finally, Steve is under ongoing informal evaluation, both within our group and by external colleagues (e.g., the Air Force Armstrong Laboratory), and we are planning formal evaluations later this year.

5 Conclusion

Steve illustrates the enormous potential in combining work in agent architectures, intelligent tutoring, and graphics. When combined, these technologies result in a new breed of computer tutor: a human-like agent that can interact with students in a virtual world to help them learn.

For more technical details on Steve, as well as a discussion on related work, see [3], [6], [7], and [8].

6 Acknowledgments

This work is a collaboration among many people. We especially thank Marcus Thiebaux and Erin Shaw for creating Steve's body, Ben Moore for his work on speech recognition, Randy Stiles and his team at Lockheed Martin for their virtual reality software and models, and Allen Munro and his team at the USC Behavioral Technology Laboratory for their simulation software and models. This work is funded by the Office of Naval Research, grant N00014-95-C-0179.

References

[1] R. Angros, Jr., W.L. Johnson, and J. Rickel. Agents that learn to instruct. In *AAAI Fall Symposium on Intelligent Tutoring System Authoring Tools*, Menlo Park, CA, 1997. AAAI Press. AAAI Technical Report FS-97-01.

[2] C. Elliott, J. Rickel, and J.C. Lester. Integrating affective computing into animated tutoring agents. In *Proceedings of the IJCAI Workshop on Animated Interface Agents: Making Them Intelligent*, pages 113–121, Nagoya, Japan, 1997.

[3] W.L. Johnson, J. Rickel, R. Stiles, and A. Munro. Integrating pedagogical agents into virtual environments. *Presence*, 1998. Forthcoming.

[4] W.L. Johnson and E. Shaw. Using agents to overcome deficiencies in web-based courseware. In J. Rickel and J. Lester, editors, *Proceedings of the AI-ED Workshop on Pedagogical Agents*, pages 48–55, Kobe, Japan, 1997.

[5] J.E. Laird, A. Newell, and P.S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.

[6] J. Rickel and W.L. Johnson. Integrating pedagogical capabilities in a virtual environment agent. In *Proceedings of the First International Conference on Autonomous Agents*. ACM Press, February 1997.

[7] J. Rickel and W.L. Johnson. Intelligent tutoring in virtual reality: A preliminary report. In B. du Boulay and R. Mizoguchi, editors, *Proceedings of the Eighth World Conference on Artificial Intelligence in Education*, pages 294–301. IOS Press, 1997.

[8] J. Rickel and W.L. Johnson. Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. 1998. Submitted for publication.

Appendix B:

Steve: An Animated Pedagogical Agent for Procedural Training in Virtual Environments

As published in
ACM SIGART Bulletin

Fall 1997

Volume 8 • Numbers 1-4 • 1997

SIGART BULLETIN

A quarterly publication of the Association for Computing Machinery Special Interest Group on Artificial Intelligence

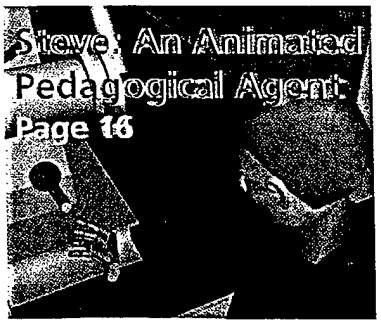
Creating Lifelike Characters in Toy Story

Page 10

ASSOCIATION FOR COMPUTING MACHINERY
1515 BROADWAY, NEW YORK, 10036

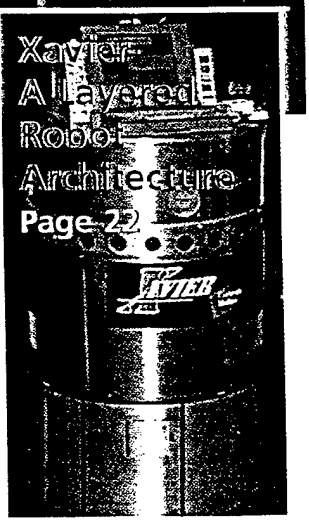
Non-Profit
Organization
U.S. Postage PAID
Easton, Maryland
Permit No. 114

Also Featuring:



Steve: An Animated Pedagogical Agent

Page 16



Xavier: A Layered Robot Architecture

Page 22

Steve: An Animated Pedagogical Agent for Procedural Training in Virtual Environments

W. Lewis Johnson and Jeff Rickel

[illegible] 90292-6695

(310) 822-1511
Fax: (310) 822-0751
johnson, rickel@isi.edu
http://www.isi.edu/isd/VET/vet.html

Virtual reality can broaden the types of interaction between students and computer tutors. As in conventional simulation-based training, the computer can watch students practice tasks, responding to questions and offering advice. However, immersive virtual environments also allow the computer tutor to inhabit the virtual world with the student. Unlike previous, disembodied computer tutors, such a "pedagogical agent" can "physically" collaborate with students, enabling new types of interaction. To illustrate the possibilities, this paper describes Steve, a pedagogical agent for virtual environments that helps students learn procedural tasks. It provides an overview of Steve's capabilities, discusses the benefits and challenges of developing Steve in the virtual [illegible]

Introduction

Virtual reality can bring simulation-based learning environments closer to real-life experience. Rather than watch the simulated world through a desktop window, students are immersed in a 3D computer simulation of their work environment, where they can improve their skills through practice on realistic tasks. Like earlier simulation technologies, virtual reality simulation environments are especially valuable in domains where real-life training is expensive or hazardous, such as surgery, air combat, and control of complex equipment. However, virtual reality can provide more realistic perceptual stimuli (e.g., visual, auditory, and haptic) than earlier technologies, thereby providing an adequate simulation for a wider range of situations [Durlach and Mavor 95]. In addition, using networked virtual reality systems, multiple students (possibly at different work sites) can learn to perform collaborative or competitive tasks together.

Virtual reality also offers exciting opportunities and challenges for intelligent tutoring systems. As in any simulation-based learning environment, students may reach impasses or fail to recognize learning opportunities, so they can benefit from a computer tutor that can answer questions and offer advice. However, immersive virtual environments also allow the computer tutor to inhabit the virtual world with the student. Unlike previous, disembodied computer tutors, such a "pedagogical agent" can "physically" collaborate with students, enabling new types of interaction.

To explore the use of intelligent tutoring systems in virtual reality, we are developing a pedagogical agent called Steve (Soar Training Expert for Virtual Environments). Steve inhabits a virtual environment, continuously monitoring the state of the environment and periodically manipulating it through virtual motor actions. He helps students learn to perform physi-

cal, procedural tasks, such as operating and repairing equipment. Steve both instructs and assists students and, later in the project, will be able to function as a team member for team training. He functions as part of a larger Virtual Environments for Training (VET) system being developed jointly by the USC Information Sciences Institute, the USC Behavioral Technology Laboratory, and Lockheed Martin.

Overview of Steve's Capabilities

The VET system allows multiple students and agents to cohabit a virtual world. Each student's interface to the virtual world is provided by special-purpose hardware and Lockheed Martin's Vista Viewer™ software [Stiles et al. 95]. Students get a 3D, immersive view of the world through a head-mounted display (HMD). Vista uses data from a position and orientation sensor on the HMD to update the student's view as they move around. Students interact with the virtual world using a 3D mouse or data gloves. Sensors on the mouse and gloves keep track of the student's hands, and the Vista software sends out messages when the student touches virtual objects. These messages are received and handled by the RIDES software [Munro et al. 93], which controls the behavior of the virtual world. (The RIDES software makes it easy for course authors to create simulation behaviors.) Currently, we are applying the VET system to training Navy personnel to operate a high-pressure air compressor (HPAC) on board a ship. The current virtual HPAC includes interactive buttons, valves, switches, gauges, indicator lights, and a dipstick. Figure 1 shows a snapshot of the virtual HPAC and its surrounding room.

Steve teaches students how to perform procedural tasks, like operating the HPAC. Our goal is to support the apprenticeship model of learning [Collins et al. 89]. This requires two capabilities: Steve must be able to demonstrate and explain tasks, and he must be able to monitor students performing tasks, providing assistance when it is needed. All of Steve's instruction and assistance is situated in the performance of domain tasks. Ideally, students should learn to apply standard procedures to a variety of situations, and they should learn the rationale behind steps in the procedures. When demonstrating, Steve performs and explains each step of the task. Steve is currently represented by a head, an upper body, and a hand that can manipulate and point at objects, as shown in Figures 2, 3, and 4. (Steve is designed to support a variety of graphical representations; we discuss a full human figure representation on page 19.) For each step in the task, Steve explains what must be done (via speech) while pointing at the objects he references, and then he performs the step.

Steve's demonstrations are not canned. Steve is given knowledge of tasks in the form of hierarchical plans. These plans include a set of steps (each either a primitive action or a subtask), a set of ordering constraints, and a set of causal links. The causal links [McAllester and Rosenblitt 91] tell Steve the role of each step in the task; each causal link states that one step (e.g., pulling out a dipstick) achieves a goal (e.g., dipstick out) that is either an end goal or a precondition for another step (e.g., checking the oil level). When demonstrating tasks, Steve continually monitors the state of the world, keeping track of whether task goals are satisfied (both end goals and intermediate goals). Using this information, Steve uses a method analogous to partial order planning [Weld 94] to keep track of which steps of the task are still relevant to completing it [Rickel and Johnson 97]. This approach to plan execution is efficient, and it forces Steve to follow standard procedures as much as possible, yet it still allows Steve to adapt the plan to unexpected events: Steve naturally re-executes parts of the plan that get unexpectedly undone, and he naturally skips over parts of the plan that are unnecessary because their goals were serendipitously achieved. Thus, unlike videos or scripted demonstrations,

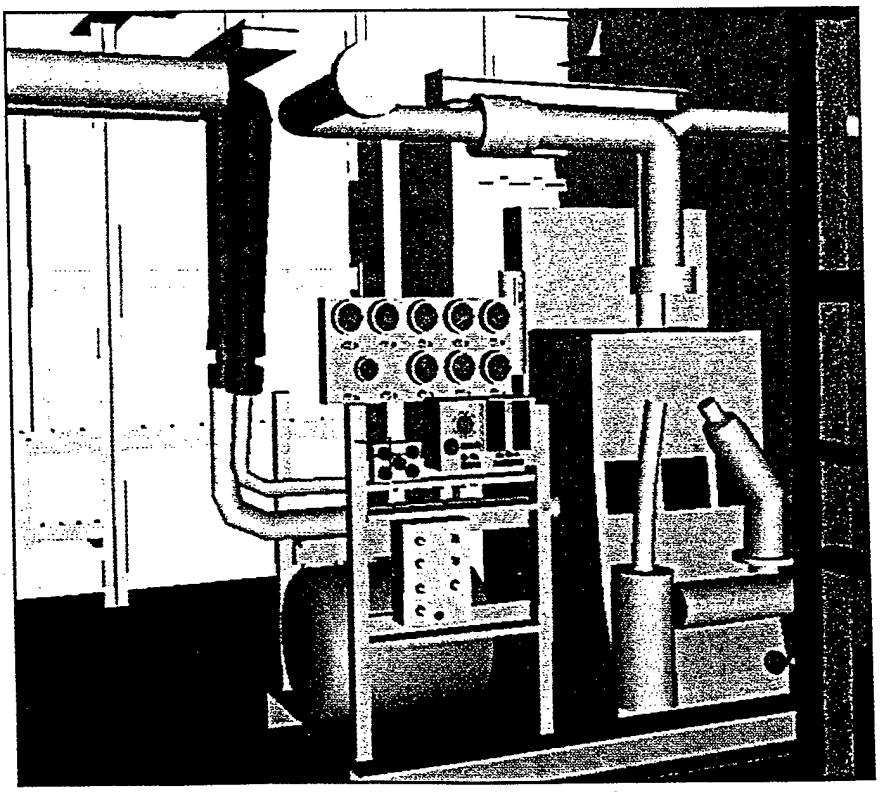


Figure 1. The virtual HPAC and part of its surrounding room.

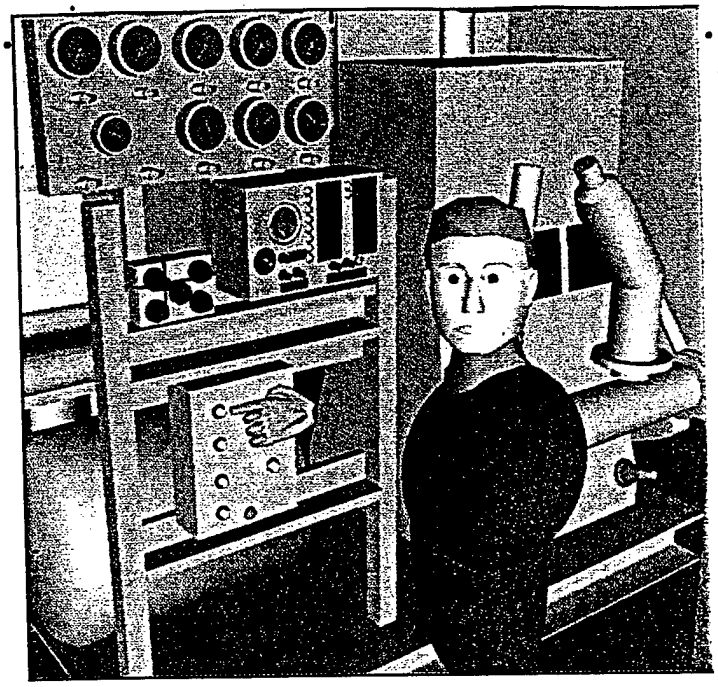


Figure 2. Steve pointing at an indicator light

Steve can adapt domain procedures to the state of the virtual world.

When demonstrating a task, Steve maintains an episodic memory of situations in which he performs actions, using Johnson's Debrief software [Johnson 94]. After the demonstration, the student can ask Steve to rationalize any of his actions. Steve recalls the situation and explains his action in terms of its relevant effect. For example, when asked why he pressed the function test button, Steve responds "That action was relevant because I wanted the drain monitor in test mode." If Steve's rationale is insufficient, the student can ask follow-up questions. For example, when asked why he wanted the drain monitor in test mode, Steve explains "That goal was relevant because I wanted to check the alarm lights." The student can continue asking follow-up questions until, ultimately, the original action has been related to an end goal of the task. Steve answers these questions using the causal links in the task model, along with his knowledge of which parts of the task were still relevant to completing it at the time of his action. Currently, students ask questions via a simple menu that appears on the palm of their hand. We are also experimenting with restricted speech recognition.

When monitoring a student performing a task, the student is in control. At any time, the student can ask Steve what to do next. When monitoring a student, as when demonstrating a task, Steve continuously monitors the state of the virtual world, mentally adapting the standard procedure for the task. Therefore, based on the current state and his own plan for completing the task, Steve can recommend an appropriate action in most situations, and he can rationalize his recommendations by answering questions as just described. The same question-answering ability is used for both demonstration and monitoring; the only difference is that monitoring questions refer to Steve's current recommendations, while demonstration questions refer to Steve's past actions.

To provide a collaborative style of interaction with the student, Steve can gracefully shift between demonstrating a task and monitoring the student's performance of that task. During Steve's demonstrations, the student can interrupt and ask to finish the task, in which case Steve shifts to monitoring. When monitoring a student, the student can always ask Steve to demonstrate a recommended action. Our aim is tosupport a natural and flexible collaboration between student and tutor.

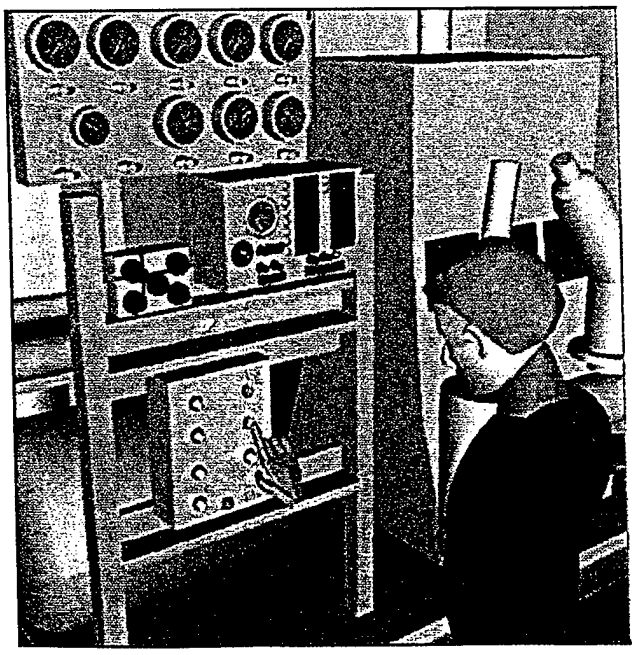


Figure 3. Steve pressing a button.

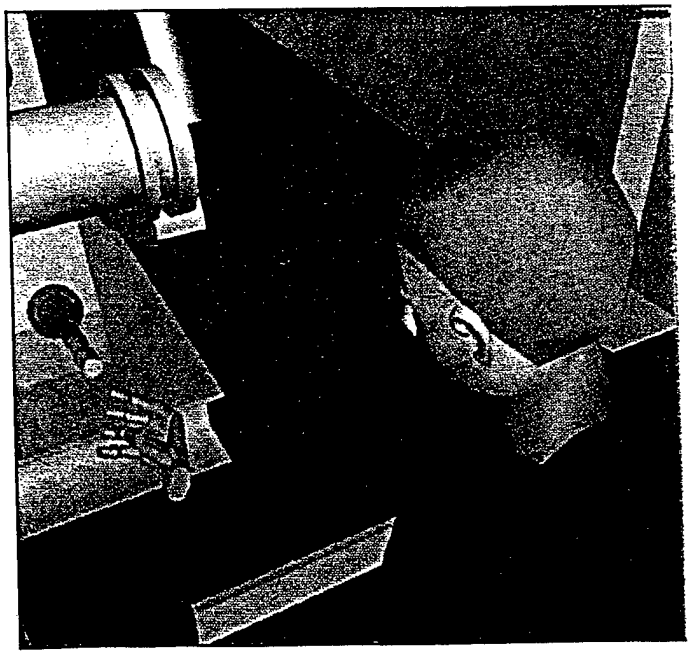


Figure 4. Steve grasping a dipstick.

Representing Steve in the Virtual Environment

Unlike most intelligent tutoring systems, Steve inhabits the virtual world along with students (and perhaps other agents). For teaching physical tasks, like operation and repair of equipment, this allows valuable interactions between Steve and students. For example, Steve can physically demonstrate task steps; this could be particularly helpful for students when those steps involve spatial motor skills. He can also draw students' attention to objects by pointing at them. Steve's embodiment will be especially useful as we extend Steve to support team training; when Steve is used to fill the role of missing team members, it will be important for other team members to keep track of Steve's activities. These types of collaboration between Steve and students would be impossible with a traditional disembodied tutor. Moreover, this approach is more flexible and interactive than video, because Steve can adapt domain procedures to the student's current situation, as described in the previous section.

While the motivation for embodiment of Steve is clear, the type of embodiment is not. Since Steve is teaching physical tasks, some variant of a human form seems most appropriate. The question is how much detail is needed. For the applications we have explored so far, a hand would probably be sufficient, and it is simple to control. Adding a head opens up additional channels of communication; for example, it allows the student to track Steve's gaze. Simple representations, such as a hand or a head and hand, are actually better than a full human figure in some respects. For example, a full human figure is more visually obtrusive, which can be a disadvantage since current head-mounted displays offer a relatively narrow field of view. Nonetheless, a human figure representation offers exciting directions for intelligent tutoring; it allows better demonstrations of physical tasks, a richer use of gestures and other types of nonverbal communication, and we expect it to be especially important as we extend Steve to team training. Therefore, we wanted an architecture for Steve that would allow us to evaluate the tradeoffs between different representations.

To experiment with different graphical representations for Steve, we designed him as two cooperating modules. The cognitive module, implemented in Soar [Laird et al. 87, Newell 90] handles the normal duties of an intelligent tutoring system, and it outputs motor commands (e.g., press a button or look at an object) to a sensorimotor module [Rickel and Johnson 97]. The sensorimotor module translates these motor commands into lower-level graphical commands to move Steve's body and, if necessary, commands to the simulator to affect the virtual world (e.g., simulate the button and its effects). We are experimenting with various graphical representations for Steve: a head, and upper body, a hand, and a full human figure. Figure 5 shows the latter two together: one Steve agent, represented by a human figure, watches (via dynamic gaze control) as another Steve agent, represented by a hand, demonstrates a task. To implement the full human figure, we use the Jack software [Badler et al. 93] developed at the University of Pennsylvania.

Jack can be used two different ways. The Jack software would allow course authors to create a variety of animation sequences, and these could be dynamically strung together by Steve during task execution. This approach is used for Stone and Lester's [Stone and Lester 96] animated pedagogical agent, although they use their own character, Herman the Bug, rather than Jack. We have focused on the alternative approach: Jack can be commanded to look at, walk to, and reach out and grasp arbitrary objects. This approach provides a finer granularity for behavior and requires much less effort by the course author, although it also gives the course author less control over motions. Our use of Jack is still rather primitive, in part because the current version of Jack makes it awkward to use another program to control him. (Jack was primarily designed for interactive control by a person via a graphical user interface.) However, work at the University of Pennsylvania has shown the potential for using Jack to autonomously move around a virtual world and manipulate objects [Geib et al. 94, Levison and Badler 94], and we believe this approach holds more promise than handcrafted animation sequences.

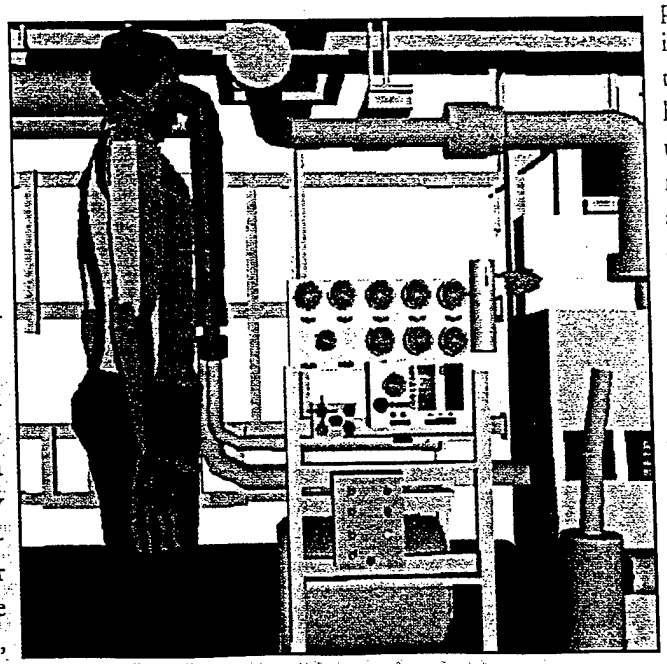


Figure 5. One Steve agent watching another

There are many useful types of nonverbal feedback that a pedagogical agent could give to a student beyond those currently used by Steve. The agent could use gaze or pointing to direct the student's attention to a new danger in the environ-

ment. He could use a nod of approval to show agreement with the student's actions, and a nod of disapproval or look of puzzlement to make the student think twice. Nonverbal feedback is also important in carrying on dialogues; for example, Cassell et al. [Cassell et al. 94] have created Jack agents that coordinate speech, intonation, facial expressions and hand gestures in conversations, albeit only with one another. Nonverbal communication provides subtle methods for influencing the student; these methods are much less obtrusive than those used by most current tutoring systems, and they should lead to more natural tutorial interactions.

While virtual reality provides exciting opportunities for human-machine interaction, it provides new challenges as well. By controlling their own field of view, students in immersive virtual environments learn to navigate around their work environment, and they can view objects from different angles. In contrast, most tutoring systems, and even multimedia presentation systems [Maybury 93], assume they can design and control the student's view. The Vista software allows Steve to control the student's field of view when necessary. However, to avoid losing the benefits of having students control their own view, we have ignored that option. Instead, Steve dynamically adapts his presentation to the student's position and orientation. For example, Steve keeps track of the objects in the student's field of view to avoid performing an action when the student is not looking. Also, when pointing at objects, Steve orients his hand based on the student's line of sight. We are working towards more natural and less constraining ways of guiding students' attention than those used by previous tutoring systems.

Status and Future Work

This paper has described Steve, an animated pedagogical agent for virtual reality. Steve inhabits a virtual world, helping students learn to perform physical, procedural tasks. Following an apprenticeship model of learning, Steve can demonstrate tasks as well as assist students while they practice. To cope with a dynamic virtual world containing other people and agents, Steve uses his knowledge of domain procedures to quickly adapt them to unexpected situations. Steve's physical presence in the virtual world enables new types of interactions with students; we expect to more fully exploit this potential in the future through richer use of nonverbal communication.

Steve has been tested on some of the operating procedures used for high-pressure air compressors aboard naval ships. We are currently extending Steve's domain knowledge to handle a wider variety of procedures on gas turbine engines, of which the HPAC is a component. As we model more of the gas turbine engine, there will be more opportunity to study tasks that require coordination of multiple team members.

We are currently extending Steve in several ways. First, to support team training, we are generalizing Steve's task representation to handle team tasks and providing ways of specifying Steve's role in a team exercise. Some Steve agents will play the role of missing team members, while others will assist students learning their role.

Second, in addition to our use of the Jack software, we plan to experiment with other human figures, such as the Otto character developed at New York University [Perlin and Goldberg 96]. Third, in collaboration with Clark Elliott of DePaul University, we are developing an emotional subsystem for Steve, to test the benefits of a more personal relationship with students. Finally, we are exploring alternative methods for teaching Steve new tasks; while a course author currently specifies new tasks textually, using our plan language, Richard Angros is developing methods by which course authors can teach Steve by demonstrating tasks in the virtual world. We believe that all these capabilities will lead to more natural and productive interactions with pedagogical agents.

We are planning a set of evaluations, within USC and by outside collaborators. A copy of Steve has been given to the Air Force Armstrong Laboratory for informal evaluation, and they will be conducting formal evaluations later in the project. We are also planning formal evaluations here at USC in conjunction with the USC School of Education. The studies at Armstrong Lab will primarily focus on the benefits of pedagogical agents for training in virtual reality, while the studies at USC will focus specifically on the benefits of believability and the use of a human figure embodiment.

Acknowledgments

The VET project is funded by the Office of Naval Research, grant N00014-95-C-0179. We gratefully acknowledge the efforts of our colleagues on the project: Randy Stiles, Laurie McCarthy, and Sandeep Tewari at Lockheed Martin, Allen Munro, Mark Johnson, Quentin Pizzini, and Jim Wogulis at the Behavioral Technology Laboratory, and Richard Angros and Erin Shaw at the USC Information Sciences Institute.

References

Badler, N.I., Phillips, C.B., and Webber, B.L., *Simulating Humans*. Oxford University Press. 1993.

Cassell, J., Pelachaud, C, Badler, N,, Steedman, M., Achorn, B., Becket, T., Douville, B., Prevost, S., and Stone, M., Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. In *Proc. ACM SIGGRAPH '94*, 1994.

Collins, A., Brown, J.S., and Newman, S.E., Cognitive apprenticeship: teaching the crafts of reading, writing, and mathematics. In L.B. Resnick, ed., *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser.* Lawrence Erlbaum Associates, 1989.

Durlach, N.I. and Mavor, A.S., editors. *Virtual Reality: Scientific and Technological Challenges.* National Academy Press, Washington, DC, 1995.

Geib, C., Levison, L., and Moore, M.B., Sodajack: An architecture for agents that search and manipulate objects. Tech. Report MS-CIS-94-16/LINC LAB 265, Dept. of Computer and Information Science, Univ. of Pennsylvania, 1994.

Johnson, W.L., Agents that learn to explain themselves. In Proc. *Twelfth National Conference on Artificial Intelligence.* AAAI Press, Menlo Park, CA., 1994, 1257–1263.

Johnson, W.L., Rickel, J., Stiles, R., and Munro, A.: Integrating Pedagogical Agents into Virtual Environments. to appear in *Presence.*

Laird, J.E., Newell, A., and Rosenbloom, P.S., Soar: An architecture for general intelligence. *Artificial Intelligence,* 1987, 33(1):1–64.

Levinson, L. and Badler, N.I., How animated agents perform tasks: Connecting planning and manipulation through object-specific reasoning. In *Toward Physical Interaction and Manipulation,* AAAI Spring Symposium Series, 1994.

Maybury, M. T., editor. *Intelligent Multimedia Interfaces.* AAAI Press, 1993.

McAllester, D. and Rosenblitt, D., Systematic nonlinear planning. In *Proc. of the Ninth National Conference on Artificial Intelligence* (AAAI-91), 1991, 634-639.

Munro, A., Johnson, M., Surmon, D., and Wogulis, J., Attribute-centered simulation authoring for instruction. In *Proc. World Conference on Artificial Intelligence in Education,* 1993, 82–89.

Newell, A., *Unified Theories of Cognition.* Harvard University Press, 1990.

Perlin, K. and Goldberg, A., Improv: A System for Scripting Interactive Actors. In *Computer Graphics Proceedings,* Annual Conference Series, ACM SIGGRAPH, 1996, 205-216.

Rickel, J. and Johnson, W.L., Integrating pedagogical capabilities in a virtual environment agent. In *Proc. of First International Conference on Autonomous Agents,* ACM Press, 1997.

Stiles, R., McCarthy, L., and Pontecorvo, M., Training studio: A virtual environment for training. In *Workshop on Simulation and Interaction in Virtual Environments,* ACM Press, 1995.

Stone, B. A. and Lester, J.C., Dynamically sequencing an animated pedagogical agent. In *Proc. of the Thirteenth National Conference On Artificial Intelligence* (AAAI-96), AAAI Press / MIT Press, 1996, 424–431.

Weld, D.S., An introduction to least commitment planning. *AI Magazine,* 1994, 15(4):27–61.